EL887747073

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Timing Synchronization Methods and Systems for Transmit Parallel Interfaces

Inventor(s):

Kun-Yung Ken Chang

Charles Huang

ATTORNEY'S DOCKET NO. RB1-038us

TECHNICAL FIELD

This invention relates to transmit parallel interfaces and, more particularly to timing synchronization methods and systems for use in connection with transmit parallel interfaces.

BACKGROUND

Transmitting certain types of data can involve transforming the data from a parallel state into a serial state for transmission. Once the data is transmitted in its serial state, it can be de-serialized into it former parallel state. For example, assume that 10 bits (in a parallel state) at 200 MHz are desired to be transmitted from a transmitter to a receiver. To effectuate transmission, the 10 bits can be converted (i.e. serialized) to individual bits and transmitted at 2.5 Gbit/second. At the receiving end, the transmitted bit stream can be converted back into its parallel state by a de-serializer that provides the data as 10 bits at 200 MHz.

The process of serializing and de-serializing data necessarily brings into play the clock domains associated with both the input data and the serializer/de-serializer. Consider, for example, Fig. 1, which shows a typical transmit parallel interface 100. On a so-called user side, parallel data 102 resides in an environment associated with a first clock domain, say clock domain A. On the serializer/de-serializer side, the data resides in an environment associated with a second clock domain, say clock domain B. Since the clocks in the two domains can be from different clock sources, the timing relationship of the two can be arbitrary. Parallel data 102 is typically received by a circuit element, such as one or more flip flops 104 that are clocked in domain B, and then provided to a serializer 106 for serialization and then subsequent transmission.

An important problem associated with the transmit parallel interface 100 pertains to how to safely transmit the data from one clock domain to another clock domain. Specifically, timing synchronization for the transmit parallel interface of high speed I/O is needed to avoid any setup time or hold time violation for the input data of the transmit parallel interface. One way of doing this is to ensure that the setup and hold window for the data is sufficient. The setup and hold time constraint requires data to be valid for certain amount of time relative to a time reference. Otherwise, the data cannot be received correctly. The implementation can affect system clocking complexity and data latency, as is known.

For example, in the past, attempts to align the clocks on both sides of the interface have been made, but have been found to be inadequate. Specifically, some implementations attempt to align the clocks on both sides of the interface by using feedback circuitry such as phase-locked loops. This solution adds clocking complexity that can be undesirable in many instances. Other implementations have attempted to use FIFO or flip flop pairs. These implementations, however, increase data latency as well as the design complexities in generating the pointer select.

Accordingly, this invention arose out of concerns associated with providing improved timing synchronization methods and systems for use with transmit parallel interfaces, that reduce clocking complexities and mitigate data latency concerns.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of an exemplary transmit parallel interface in accordance with the prior art.

Fig. 2 is a timing diagram of the Fig. 1 transmit parallel interface.

Fig. 3 is a block diagram of an exemplary transmit parallel interface in accordance with one embodiment.

Fig. 4 is a timing diagram of the Fig. 3 transmit parallel interface.

Fig. 5 is a flow diagram that describes steps in a method in accordance with one embodiment.

DETAILED DESCRIPTION

Overview

In various embodiments described below, a transmit parallel interface is provided in which the input data is first clocked by a system clock which is provided by the user to acquire the data without timing errors and then an internal clock signal is generated that maximizes the timing window to be used to receive the reclocked data. In some embodiments, this is done by providing a clock signal (designated as "TclkPar") in one clock domain that has a rising edge located close to the falling edge of a system clock in another clock domain. This helps to reduce error from one clock domain to the other clock domain.

Exemplary Embodiment

Fig. 2 shows a timing diagram of a transmit parallel interface in which an ideal TclkPar signal is generated. A system clock 200 designated as "SystemClk" represents a clock used by the input data (i.e. parallel data) in the user's clock domain. A high speed clock 202, designated "Tclk," is provided and is used to clock a transmitter that transmits serialized data. Typically, Tclk is *N* times faster than SystemClk. TclkPar 204 constitutes the divided clock of Tclk for the parallel

interface and has, in this example, the same frequency as the SystemClk. TclkPar is the clock in a second clock.

In the embodiments described below, the system clock (SystemClk 200) is at a dividable frequency of Tclk. More specifically, there should be no frequency deviation between SystemClk and TclkPar. In addition, the divide ratio between the clocks should be no less than four. However, the same idea can be applicable to the case where the frequency of SystemClk is half, one quarter, or ($1/2^N$) of the frequency of TclkPar.

In Fig. 2, there are five Tclk rise edges within a SystemClk cycle. In this example, the clock-divide-ratio is five between Tclk and SystemClk. TclkPar, which rises after a clock-to-q delay from one of the five Tclk rise edges, is chosen from the Tclk edge that is closest to the falling edge of SystemClk. Assuming the clock-to-q delay is zero in all the flip-flops involved, this clock placement of TclkPar can tolerate most timing variations due to clock path mismatches and clock jitter. In reality, where the clock-to-q delay is nonzero, one can add the same delay in some of the paths to cancel its effect.

Exemplary Circuit Implementation

Fig. 3 shows an exemplary transmit parallel interface circuit implementation 300 that generates a desirable TclkPar signal that can ensure that the setup and hold window of the input data is maximized.

In the described embodiment, circuit 300 is desirably fabricated on a single chip, although such need not be the case. In the illustrated example, TclkPar is generated by a clock divider (1/N) using Tclk. Since TclkPar is generated by a

clock divider using Tclk, one can select one of N >= 4 Tclk to place TclkPar around the falling edge of the SystemClk.

In the illustrated example, circuit 300 includes flip flop assemblies 302, 304, 306 and 308. The flip flop assemblies can include one or more flip flops. A clock divider circuit 310 is provided, as is a serializer circuit 312. Standard clock divider and serializer circuits can be used as will be appreciated by those of skill in the art.

One goal of the circuit about to be described is to generate a clock signal (in this case TclkPar) having a rising edge that is placed in a desirable location so that data can be safely sent from one clock domain to another. Recall that the input data has some particular relationship to the system clock in the first domain. In our example, the clock in the first domain is the SystemClk. Circuit 300 then places the second domain clock (i.e. TclkPar) at the right place so that the input data can be sent safely from one domain to another.

Specifically, we know, in this example, that within one SystemClk cycle, the best place to place the second domain clock (TclkPar) is on or around the falling edge of the SystemClk. This helps to ensure that the reclocked input data can be clocked by TclkPar with maximum tolerance on clock uncertainty. It is to be appreciated that this specific clock placement constitutes but one example of where the second domain clock can be placed. Accordingly, it is possible to use the principles described in this document to place the second domain clock in a position other than the falling edge of the SystemClk without departing from the spirit and scope of the claimed subject matter.

Referring still to Fig. 3, input data in a parallel state is received by flip flop assembly 302 and clocked by SystemClk to produce clocked input data signal

Din_sclk. A reset signal is received by flip flop assembly 304 and is clocked by SystemClk to produce a clocked reset signal Reset_sclk. The clocked reset signal Reset_sclk carries information pertaining to the system clock SystemClk. The clocked input data signal Din_sclk is provided to flip flop assembly 306 where it is clocked by the second clock domain signal TclkPar.

The way that TclkPar is produced, in this example, is as follows. Recall that it is highly desirable, in this specific instance, to place TclkPar right on or very close to the falling edge of the system clock (SystemClk). Since there are five clock edges of Tclk within one clock cycle of the system clock SystemClk (see Fig. 2 where the five clock edges are encircled), there are conceivably five locations to choose from to align TclkPar. Accordingly, generating TclkPar just depends on how counter clock divider circuit 310 is reset. In this example, the clocked reset signal Reset_sclk is provided to flip flop assembly 308 and clocked by Tclk to produce a reset_q signal that is in the Tclk domain. The reset_q signal provides information about where the rising edge of SystemClk is located.

Clock divider circuit 310 receives the reset_q signal at a reset input (designated "reset") and is clocked by Tclk to generate and place TclkPar based on the reset_q signal. The clocked input data signal Din_sclk is thus reclocked by TclkPar and serialized by serializer 312 for transmission as serial data.

To reduce the possibility of metastability, flip flop assembly 308 should be provided as a series of flip flips. Additionally, to further maximize the setup and hold window, the Reset_sclk and Din_sclk signals should be matched. This can cancel the clock-to-q delay of flip flop assemblies 302 and 304, associated wire delay and any added circuitry between flip flop assemblies 302 and 306.

Fig. 4 shows the timing diagram of circuit 300. Note that Reset_sclk and Din_sclk have the same timing as indicated at 400. In the described embodiment, only one constraint is imposed on SystemClk: it needs to be of a dividable frequency of Tclk. There is no constraint on its phase. In other words, one does not need to align SystemClk to any other reference clock. In addition, since no FIFO is needed, the latency is not compromised. Based on Reset_sclk, the 1/N circuit 310 in Fig. 3 places TclkPar from the Tclk edge which is closest to the falling edge of SystemClk. As a result, there is plenty of timing between the Din_sclk transition edges and TclkPar rising edge. Hence, Din_sclk can be safely clocked by TclkPar and sent to Serializer 312 in Fig 3.

Exemplary Methods

Fig. 5 shows a flow diagram that describes steps in a method in accordance with one embodiment. In the illustrated and described embodiment, the steps are implemented in digital circuitry. One specific example of digital circuitry is given above in Fig. 3.

Fig. 5 is a flow diagram that describes steps in a method in accordance with one embodiment. In the illustrated and described embodiment, the steps are implemented in digital circuitry.

Step 500 provides a system clock in a first clock domain and clocks input data using the system clock. The system clock frequency is usually set by the user and should be about the same frequency as used by user's core logic. In the Fig. 3 example, the system clock is referred to as SystemClk. In the Fig. 3 example, the reclocked input data is provided as Din_sclk. Step 502 provides a high speed clock that is N times faster than the system clock. In the Fig. 3 example, the high

speed clock is referred to as Tclk. Step 504 provides a reset signal and step 506 clocks the reset signal with the system clock to provide a clocked reset signal. In the Fig. 3 example, this step is implemented by flip flop assembly 304 which receives the reset signal and is clocked by the system clock to produce the Reset_sclk signal. Step 508 clocks the clocked reset signal (Reset_sclk) with the high speed clock (Tclk) to produce an output signal. This step is implemented, in the Fig. 3 example, by flip flop assembly 308 which receives the Reset_sclk signal and clocks it to produce an output signal referred to as the reset_q signal.

Step 510 uses the output signal (reset_q) and the high speed system clock to generate a clock signal in a second clock domain. In the Fig. 3 example, the generated clock signal is referred to as "TclkPar". This step is implemented in the Fig. 3 example through the use of a clock divider circuit or counter 310 that is reset using the output signal reset_q so that it selects an appropriate edge of the high speed clock relative to which to generate the second clock domain signal (TclkPar). Step 512 then clocks the input data using the second clock domain signal and step 514 serializes the clocked data for transmission.

Conclusion

The embodiments described above can provide improved timing synchronization methods and systems for use with transmit parallel interfaces that reduce clocking complexities and mitigate data latency concerns.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or

steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.